Training Text

マイコントレーニングボード
MT-R300

ステッピングモーター基板（MT-E506）編

VPort Lab. 代表
元長野県工科短期大学校教授
工学博士 星野　俊行　著

Sunhayato

安全上のご注意

このたびは弊社製品をご使用いただき、誠にありがとうございます。本項では、誤った取り扱いによる事故を未然に防ぐための安全上の注意事項を説明しています。弊社製品をご使用になる前に必ずお読みください。

⚠警告	この表記を無視して誤った取り扱いをすると、死亡や重傷など、人体への重大な障害をもたらす恐れのある内容について示しています。
⚠注意	この表記を無視して誤った取り扱いをすると、軽傷または中程度の障害をもたらす恐れのある内容について示しています。また、本製品や本製品に接続している機器に損傷を与える可能性がある事項についても示しています。

⚠警告

水場禁止	▶水分の多いところ、水がかかる場所では、本製品は使用しないでください 風呂場や台所など水分の多いところ、水がかかる場所では、本製品は使用しないでください。 火災、感電、故障の原因となります。
禁止	▶医療、軍事、航空宇宙、列車、運送、原子力などの制御設備へは使用しないでください 医療機器、軍事機器、航空宇宙機器、運送、原子力などの制御設備などの人命に関わるシステムへの使用は意図しておりません。
接触禁止	▶雷が鳴りはじめたらご使用をお控えください 近くに雷が発生したときは、パソコンから本製品を抜いてご使用をお控えください。また、ご使用のパソコンの取扱説明書に従って、電源プラグをコンセントから抜くなどしてください。雷によっては、火災、感電、故障の原因となることがあります。

⚠注意

プラグの差し込み	▶プラグは確実に差し込んでください 差し込みが不完全ですと火災、感電、過熱、故障の原因になります。
発火注意	▶発火、発煙、異臭への対処 発火、発煙、異臭がするなどの異常がありましたら使用を直ちに中止してください。そのまま使用すると、火災、故障の原因となります。 すぐにパソコンから抜き、煙などの異常が出なくなるのを確認し、販売店などに修理をご依頼ください。
分解禁止	▶分解・改造しないでください 分解、改造しないでください。怪我、感電、故障の原因となります。本製品の分解、改造による怪我や事故について、当社は責任を負いかねます。
接触禁止	▶濡れた手での操作は避けてください 濡れた手で電源ケーブル・プラグを抜き差ししないでください。また、製品に触れないでください。感電の原因となることがあります。

⚠注意

禁止	▶以下のような場所では使用しないでください 本製品を以下のような場所で使用すると、動作不良、故障の原因となります。 ・振動や衝撃が加わる場所 ・直射日光のあたる場所 ・湿気やホコリが多い場所 ・温度差の激しい場所 ・熱を発生するもの（暖房器具など）の近く ・強い磁力、電波が発生するもの（磁石、ディスプレイ、スピーカー、ラジオ、無線機など）の近く ・湿気の多い場所
子供注意	▶子供の手の届かない場所に置いてください 本製品に装着されている電子部品など子供が飲み込まないように注意してください。
引抜禁止	▶通信中はケーブルを引抜かないでください 本製品がパソコンと通信中の場合はケーブルを引抜かないでください。ケーブルの引抜きは、必ず通信していないときに行ってください。故障の原因になることがあります。
安全設計	▶安全設計をしてください 本製品を、高度な信頼性を必要とするシステムに使用する場合は、冗長設計、誤動作防止設計など充分な安全設計を必ず行ってください。本製品の故障、傷害により生じるいかなる損害、事故について当社は責任を負いかねます。
保管注意	▶長期間使用しない場合の保管について 長期間使用しない場合は、帯電防止袋などに入れ、ホコリなどが入らないようにしてください。ホコリや汚れが付着すると短絡、接触不良などの原因になります。
ホコリ注意	▶製品の清掃について 製品にホコリや汚れなどが付着すると短絡、故障の原因になりますので、下記の「▶お手入れについて」に従って清掃してください。
薬品注意	▶お手入れについて ホコリが付着した場合はサンハヤト製ジェットブロー（JBK-480）などのガススプレーで吹き飛ばしてください。
使用注意	▶故障、破損時の処理について 本製品が故障もしくは破損した場合は、速やかに使用を中止してください。そのまま使用しますと火災、感電、怪我の原因になるおそれがあります。
廃棄注意	▶本製品の廃棄について 本製品の廃棄は、各自治体の廃棄ルールに従ってください。詳しくは各自治体にお問い合わせください。

本資料についてのご注意

本資料について

・本資料は、電子工作や電子回路、パーソナルコンピュータの操作について一般的な知識をお持ちの方を対象にしています。
・本資料を元に操作するには、株式会社ルネサス エレクトロニクス製 H8/300H マイコンについての知識や開発環境などが必要です。
・Microsoft®、Windows® は米国 Microsoft 社の米国およびその他の国における登録商標です。
・その他、記載されている会社名、製品名は各社の商標または登録商標です。

本資料のご利用にあたって

・この取扱説明書に掲載している内容は、お客様が用途に応じた適切な製品をご購入頂くことを目的としています。その使用により当社及び第三者の知的財産権その他の権利に対する保証、又は実施権の許諾を意味するものではありません。また、権利の侵害に関して当社は責任を負いません。
・本資料に記載した情報を流用する場合は、お客様のシステム全体で充分評価し適用可能かご判断願います。当社では適用可能判断についての責任は負いません。
・本資料に記載してある内容は、一般的な電子機器（学習教材、事務機器、計測機器、パーソナル機器、コンピュータ機器など）に使用されることを目的としています。高い品質や信頼性が要求され、故障や誤作動が直接人命を脅かしたり人体に危害を及ぼす恐れのある、医療、軍事、航空宇宙、原子力制御、運輸、移動体、各種安全装置などの機器への使用は意図も保証もしておりません。
・この取扱説明書の一部、又は全部を著者および当社の承諾なしで、いかなる形でも転載又は複製されることは堅くお断りします。
・全ての情報は本資料発行時点のものであり、当社は予告なしに本資料に記載した内容を変更することがあります。
・この資料の内容は慎重に制作しておりますが、万一記述誤りによってお客様に損害が生じても当社はその責任を負いません。
・本資料に関してのお問合せ、その他お気付きの点がございましたら、当社までお問合せください。
・本資料に関する最新の情報はサンハヤト株式会社ホームページ（http://www.sunhayato.co.jp/）に掲載しております。

このテキストについて

本テキストでは、サンハヤト H8 マイコントレーニングボード MT-R300 のプログラム開発を、HEW（High-Performance Embedded Workshop）を使用して行った場合について説明しています。HEW の詳細な使用方法や注意事項につきましては、ルネサス エレクトロニクス社発行のマニュアルを参照してください。

サンハヤト H8 マイコントレーニングボード MT-R300 概要

- ターゲットマイコン　：H8/300H シリーズ 3062 グループ HD64H3062BF（以下 H8/3062BF マイコン）
- 書き込みボード　　：MT-R300（H8/3062BF マイコン モード 5（内蔵 ROM 有効拡張 16M バイトモード）、モード 7（シングルチップアドバンストモード）対応）

開発に必要なもの

- 開発用 PC（以下の条件を満たすもの）

PC 本体	Pentium Ⅲ 600MHz 以上を搭載した IBM PC/AT 互換機
OS	Windows XP, Windows Me, Windows 98SE, Windows 2000（Windows XP 推奨）
メモリ	128MB 以上（ロードモジュールの 2 倍以上）
ハードディスク	エミュレータ用ソフトウェアのインストールには 100MB 以上の空き容量が必要
入力デバイス	マウス、またはマウス相当のポインティングデバイス
インターフェイス	1 ポート以上の USB インターフェイス

- サンハヤト H8 マイコントレーニングボード　MT-R300
- 付属の USB ケーブル

このマニュアルで使用しているツール類

このスタートアップガイドでは、以下の OS、ツールのバージョンで動作したものとして説明しています。

- ホスト PC の OS　　：Windows XP Professional
- 統合化開発環境　　：HEW V.4.03.00001
- C コンパイラ　　：H8S,H8/300 Standard Toolchain (V.6.02 Release00)
- フラッシュ書き込みツール　：FDT（FlashDevelopmentToolKit ）Ver4.02

HEW のバージョンは、HEW の「 ヘルプ 」メニューの「High-Performance Embedded Workshop のバージョン情報 」で確認できます。

このマニュアルで紹介している各社サイト、ツールについて

本スタートアップガイドで紹介している各社サイトやツールは、本スタートアップガイド発行時のものを掲載しております。各社サイトの構成やツールのバージョン、またバージョンに伴ってツールのファイル名などが変わっている場合がありますのでご了承ください。

目　次

このテキストについて …… 5

サンハヤト H8 マイコントレーニングボード MT-R300 概要 …… 5
開発に必要なもの …… 5
このマニュアルで使用しているツール類 …… 5
このマニュアルで紹介している各社サイト、ツールについて …… 5

1. はじめに …… 7

2. 基礎知識 …… 8

2.1　ステッピングモーター …… 8
2.2　フォトインタラプタ …… 12

3. 接続図の概要 …… 13

4. 基本実習 …… 14

4.1　ユニポーラ駆動 …… 14
実習 4.1.1a　ユニポーラー駆動の 1 相励磁でモーターを左（右）に回転する …… 14
実習 4.1.1b　ユニポーラー駆動の 1 相励磁でモーターを左（右）に回転する（引数を使わない方法） …… 15
実習 4.1.2a　ユニポーラー駆動の 2 相励磁でモーターを左（右）に回転する …… 16
実習 4.1.2b　ユニポーラー駆動の 2 相励磁でモーターを左（右）に回転する（引数を使わない方法） …… 17
実習 4.1.3a　ユニポーラー駆動の 1-2 相励磁でモーターを左（右）に回転する …… 18
実習 4.1.3b　ユニポーラー駆動の 1-2 相励磁でモーターを左（右）に回転する（配列を検索する方法） …… 18
実習 4.1.3c　ユニポーラー駆動の 1-2 相励磁でモーターを左（右）に回転する（演算で求める方法） …… 19
4.2　ユニポーラ駆動関数のライブラリー作成 …… 21
実習 4.2.1　ユニポーラー駆動の 1 相励磁でモーターを左（右）に回転する（ライブラリー使用） …… 21
実習 4.2.2　ユニポーラー駆動の 2 相励磁でモーターを左（右）に回転する（ライブラリー使用） …… 22
実習 4.2.3　ユニポーラー駆動の 1-2 相励磁でモーターを左（右）に回転する（ライブラリー使用） …… 23
実習 4.2.4　ユニポーラー駆動の 1-2 相励磁で原点復帰し、SW2 で左（右）回転し、SW1 で停止する …… 24
4.3　バイポーラ駆動 …… 26
実習 4.3.1a　バイポーラー駆動（2 相励磁）で左（右）に回転する …… 26
実習 4.3.1b　バイポーラー駆動（2 相励磁）で左（右）に回転する（配列を検索し次の値を求める） …… 26
実習 4.3.1c　バイポーラー駆動（2 相励磁）で左（右）に回転する（演算で求める方法） …… 27
実習 4.3.2　バイポーラー駆動（2 相励磁）で左（右）に回転する（ITU0 のトグル出力機能を使う方法） …… 29
4.4　バイポーラ駆動関数のライブラリー作成 …… 31
実習 4.4.1　バイポーラー駆動（2 相励磁）で左（右）に回転する（ライブラリー使用） …… 31
実習 4.4.2　原点復帰し、SW2 で左（右）に回転し、SW1 で停止する（ライブラリー使用） …… 33
実習 4.4.3　原点復帰し、SW2 で左（右）に回転し、指定位置に達したら停止する
（ITU0 を用いる方法） …… 34

5. おわりに …… 36

1. はじめに

ステッピングモーターは、複数のコイルに順番にパルス電流を流すことで一定の角度だけ回転するもので、パルスモーターとも言われます。これは制御対象の位置決めを正確に行うことができるため、産業用ロボット、X-Y テーブル、プロッタ、プリンタ、FAX、他に組み込まれています。制御技術者はこの知識が必要不可欠です。

このボードはステッピングモーター、モーターの原点位置を検知する赤外線センサ（フォトインタラプタ）、モータのドライバー IC（ユニポーラ駆動用とバイポーラ駆動用）、押しボタンスイッチ 2 個、他を搭載しています。これを用いてモーターのハードウエアの概要を学習し、その回転位置、回転速度を制御するプログラムを作成します。

搭載されているモーターは 2 相のユニポーラ型（結線 :6 本）で、そのままユニポーラ駆動できます。この 6 本線中のプラス電源線（+Vs）2 本を未接続にすることで、2 相のバイポーラ型（結線 :4 本）に変身させて、バイポーラ駆動ができます。

このモーターはドーターボード（モーター基板）上に設置され、コネクタを介してマザーボードに接続されています。このモーター基板（コネクタ）を引き抜いて 180 度回転して差し替えることで、マザーボード上の対応するドライバー IC に接続されるため、2 つの駆動方式を切り替えて実習ができます。これがこのボードの大きな特徴です。

このモーターは DC5V、80mA/ 相のため、VPort のバスパワーで十分動作します。赤外線センサがモーター円板の穴位置を検知することで、モーターを原点復帰できます。モニタ LED で、モーターの制御信号、センサー信号、押しボタンスイッチ信号のレベル（H/L）をモニターできます。各信号の変化を視認できるため、より効果的な教育ができます。

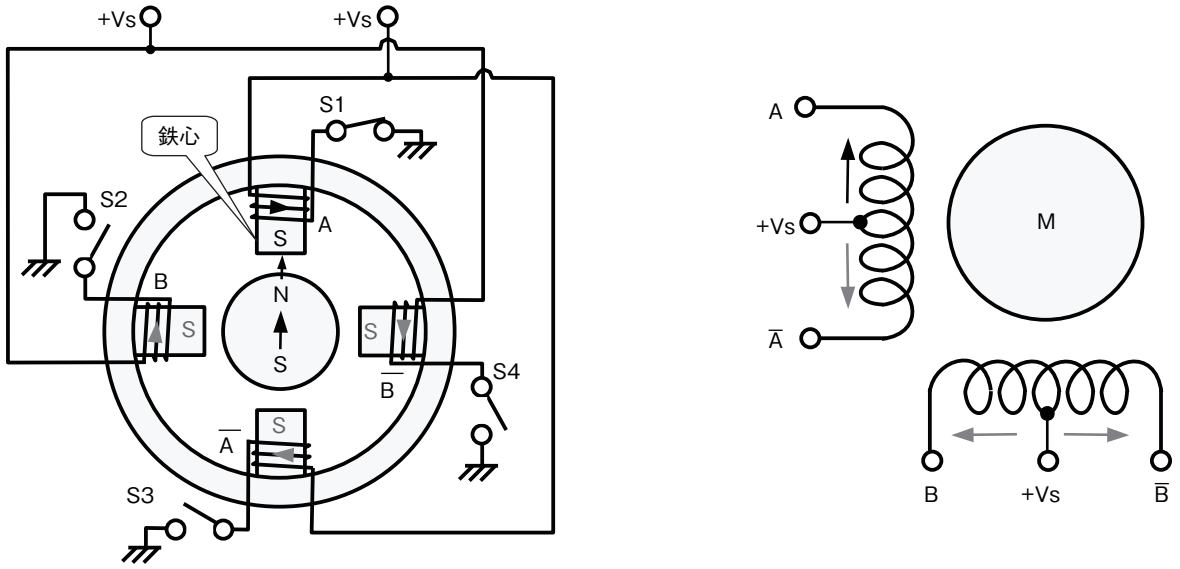
2. 基礎知識

2.1 ステッピングモーター

ステッピングモーターは、コイルに順番にパルス電流を流すことで回転します。コイルの構成（相数）は 2 相のものが一般的ですが、3 相、5 相のものもあります。回転部の構造から、パーマネント・マグネット型（PM 型）、可変リラクタンス型（VR 型）、ハイブリッド型（HB 型）に分けられます。このボードに搭載されているモーターは 2 相のパーマネント・マグネット型です。

（1）駆動方式による分類

①ユニポーラ型



（a）構造と原理　　（b）図記号

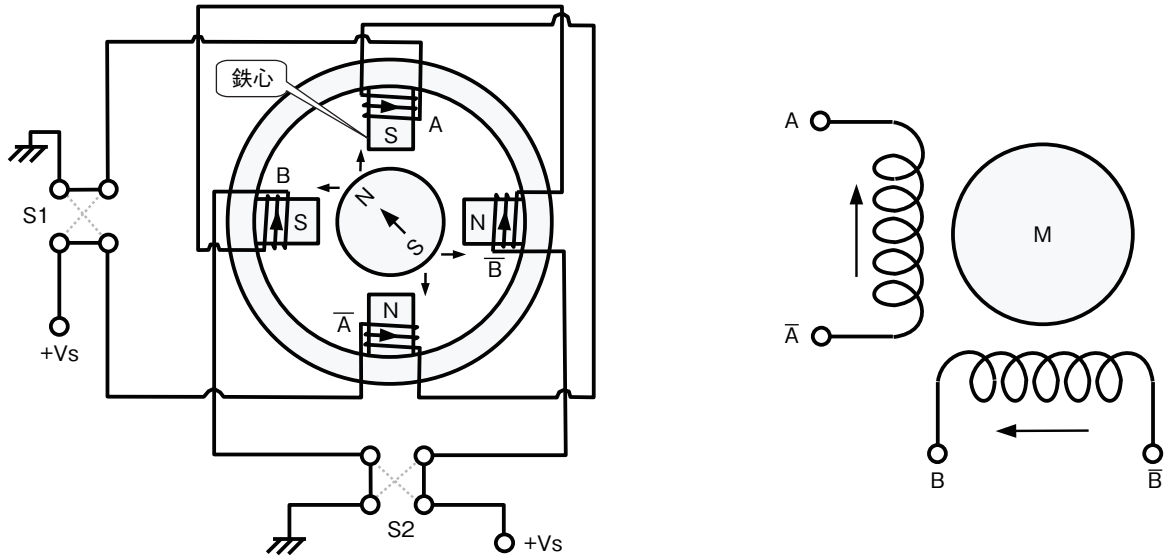
図 2-1　ユニポーラ型ステッピングモータの概要

図 2-1 は、標準的な 2 相ユニポーラ型ステッピングモーター（以下「モーター」という）の概要を示します。(a) が構造と原理です。回転軸に取り付けられた永久磁石（ロータ：回転子）と、その外周部に配置した 4 個のコイル（ステータ：固定子）で構成されています。2 つの +Vs 端子に + 電源を接続します。これは、電源の極性が変わらないので、ユニポーラ（単極）型といいます。

スイッチ S1 をオンにするとコイル A に電流が流れ、コイルが巻かれた歯車状の鉄心先端（歯車）が S 極に励磁され、回転子の N 極が上に引き寄せられます。次に S2 をオンにするとコイル B に電流が流れ、同様に歯車が S 極に励磁され、回転子の N 極が左に引き寄せられ、左に 90 度（ステップ角）回転します。続いて、S3、S4 と切り替えることで、それぞれコイル $\overline{A}$、$\overline{B}$ 相が励磁され、回転子が下、右の順に引き寄せられ回転します。この切り換え動作を高速で繰り返すことで、モーターが連続して左回転します。逆順にスイッチを切り換えることで、右回転します。この切り換え速度により、回転速度が決まります。

(b) が図記号を示します。コイル A、$\overline{A}$ の +Vs を 1 つにまとめて共通端子とし、それらを 1 つのコイルとして表示します。同様に B、$\overline{B}$ も 1 つのコイルとして表示します。外部への配線数は 6 本です。2 相（4 つ）のコイルで構成されるため 2 相のモーターと呼びますが、4 相と呼ばれることもあります。

②バイポーラ型



(a) 構造と原理　　(b) 図記号

図 2-2　バイポーラ型ステッピングモータの概要

図 2-2 は、標準的な 2 相バイポーラ型モーターの概要を示します。(a) が構造と原理です。構造はユニポーラ型とほぼ同じですが、直列接続された 2 つのコイルの中間の +Vs 端子がありません。そのため、逆回転させるときはコイルの両端に供給する電源の極性を逆にします。そのため、バイポーラ（双極）型といいます。

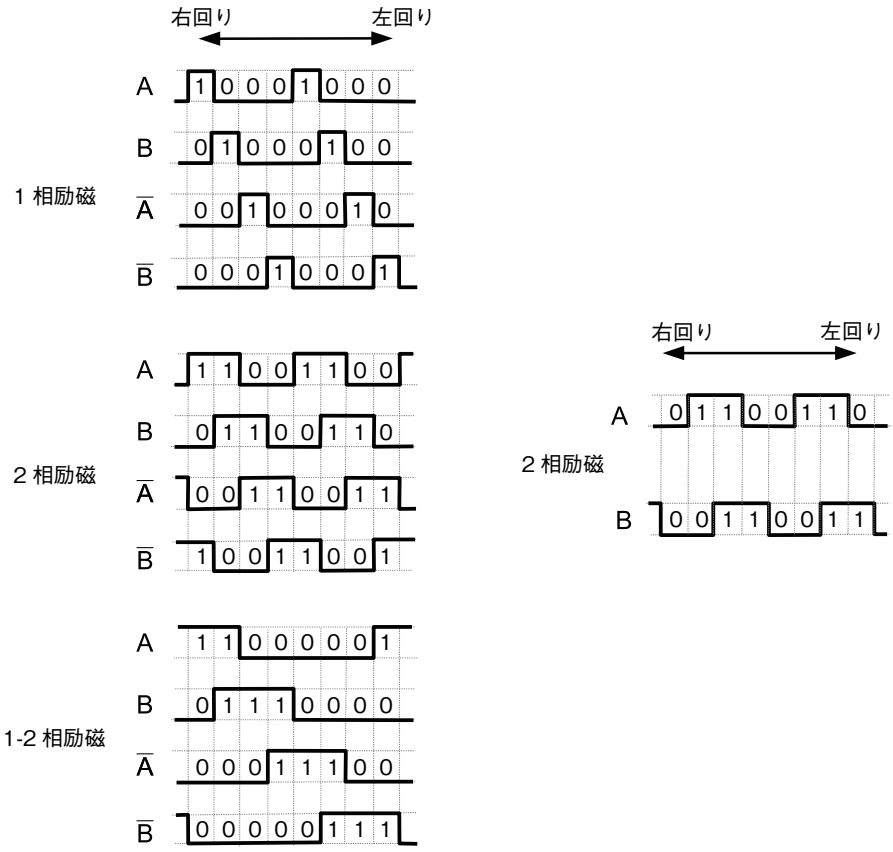
スイッチ S1 をストレート結線（実線）にすると、コイル $\overline{A}$、A の順に電流が流れ、歯車先端がそれぞれ N 極、S 極に励磁され、回転子の N 極が上（S 極が下）に引き寄せられます。そのスイッチをクロス結線（破線）にすると、コイル A、$\overline{A}$ の順に逆向き電流が流れ、歯車先端がそれぞれ N 極、S 極に励磁され、回転子の N 極が下（S 極が上）に引き寄せられます。同様に、スイッチ S2 をストレート結線（実線）にすると、コイル $\overline{B}$、B の順に電流が流れ、歯車先端がそれぞれ N 極、S 極に励磁され、回転子の N 極が左（S 極が右）に引き寄せられます。そのスイッチをクロス結線（破線）にすると、コイル B、$\overline{B}$ の順に逆向き電流が流れ、歯車先端がそれぞれ N 極、S 極に励磁され、回転子の N 極が右（S 極が左）に引き寄せられます。

S1、S2 はストレート、クロスの結線を選択するもので、オフにはできません。従って、2 相（4 つ）のコイルには常にいずれかの向きに電流が流れています。S1 をストレート結線、S2 をストレート結線にすると、回転子（N 極）は、上、左の 2 方向に引き寄せられ、ベクトル合成した左斜め上 45 度を向きます。次に、S1 をクロス結線、S2 をストレート結線にすると、回転子（N 極）は、下、左に引き寄せられ、左に 90 度回転し、左斜め下 45 度を向きます。続いて、S1 をクロス結線、S2 をクロス結線にすると、左に 90 度回転し、右斜め下 45 度を向きます。S1 をストレート結線、S2 をクロス結線にすると、左に 90 度回転し、右斜め上 45 度を向きます。この切り換え動作を高速で繰り返すことで、モーターが連続して左回転します。逆順にスイッチを切り換えることで、右回転します。この切り換え速度により、回転速度が決まります。

（b）が図記号を示します。+Vs 端子がないため、外部への配線数は 4 本です。

図 2-1、2-2 ではステップ角が 90 度で、4 ステップで 1 回転することになります。実際のモーターは歯車数を多くしてステップ角をもっと小さくしています。MT-E506 に搭載されているモーターはパーマネント・マグネット型で、ステップ角が 18 度で 20 パルスで 1 回転します。細かい制御が必要な場合には、ステップ角が 1.8 度（200 パルスで 1 回転）程度のハイブリッド型が適切です。

(2) 励磁方式



(a) ユニポーラ駆動　　(b) バイポーラ駆動

図 2-3　励磁方式と電流のタイミングチャート

① 1 相励磁

図 2-3 にそれぞれの励磁方式と電流のタイミングチャートを示します。ユニポーラ駆動の場合、図 2-1 のように 1 つのコイル(1 相)に電流を流し(励磁し)、モーターを回すことを学習しました。これを 1 相励磁といいます。バイポーラ駆動の場合はこの励磁方法はできません。

② 2 相励磁

バイポーラ駆動の場合、図 2-2 のように 、常に 2 つのコイル（2 相）に電流を流し（励磁し)、モーターを回すことを学習しました。ユニポーラ駆動の場合も、同様に 2 つのコイル（2 相）に電流を流し（励磁し)、モーターを回すことができます。これらを 2 相励磁といいいます。これは 、1 相励磁と比べ、2 つのコイルで引き寄せるため、大きなトルクが得られ、回転が安定しますが、消費電力は 2 倍になります。

③ 1-2 相励磁

図 2-3 のように、1 相励磁と 2 相励磁を交互に繰り返すと、ステップ角が半分になります。これを 1-2 相励磁といいます。この方式は他の方式と比べ細かい制御ができます。バイポーラ駆動の場合はこの励磁方法はできません。

(3) モーターの駆動回路

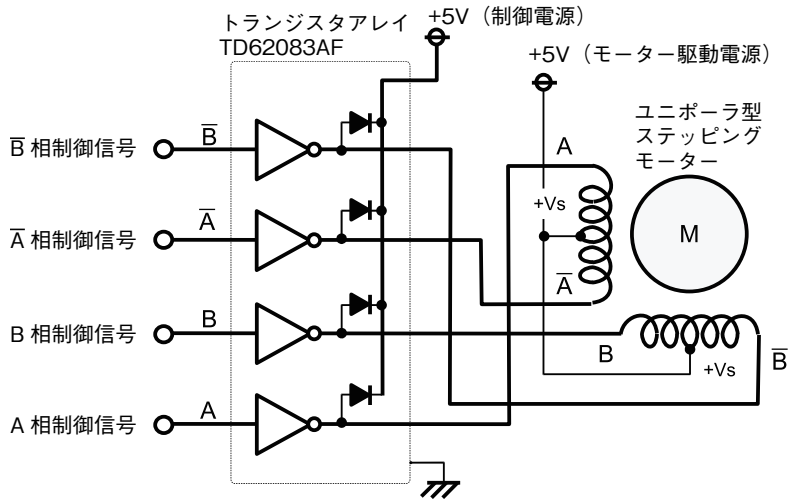
① ユニポーラ駆動

図 2-4 はモーターの駆動回路を示します。(a) はユニポーラ駆動の場合です。+Vs 端子にモーター駆動用電源を接続します。トランジスター（エミッタ接地回路）を用いてモーターコイルの電流を制御（オン／オフ）します。ここでは個別部品（トランジスター）の代わりに、複数個のトランジスター回路をまとめたトランジスタアレイ（東芝：

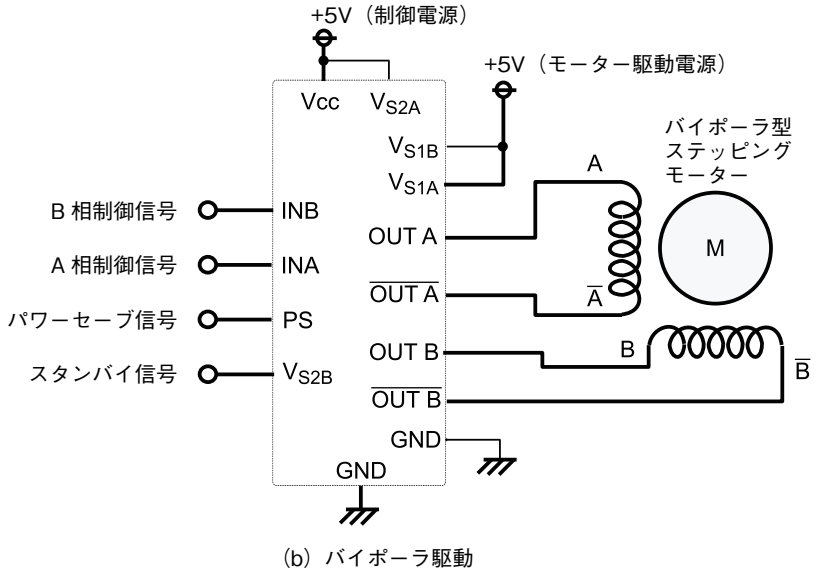
TD62083AF）を用います。内蔵されたダイオードにより、コイルに生じる逆起電力を吸収します。4 つの制御信号により、1 相、2 相、1-2 相に電流を流し、モーターを制御します。

② バイポーラ駆動

図 2-4 の（b）はバイポーラ駆動の場合です。バイポーラ型モーター用ドライバー IC（東芝 :TA7774FG）を用いて、モーターコイルに流す電流の向きを制御します。この内部のブリッジドライバーにより電流の向きを変えることができます。制御回路に +Vcc（+5V）を供給し、モーター駆動回路に高電圧 +Vs を供給します。ここで使われているモーターは +5V 仕様のため、+Vs 電源も +5V に接続します。IN A、IN B の制御信号により、2 相の電流の向きを切り替えてモーターを制御します。VS2B はスタンバイ信号（0: スタンバイ、1: 動作）、PS はパワーセーブ信号（0: 通常、1: パワーセーブ）です。



（a）ユニポーラ駆動



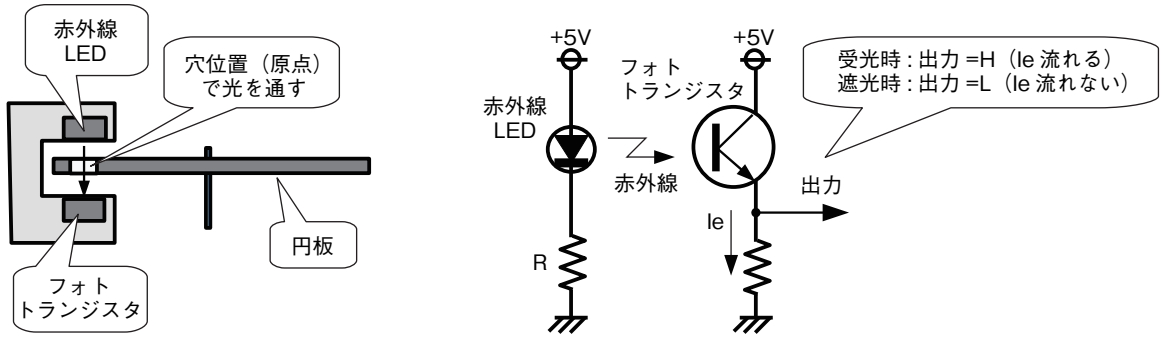
（b）バイポーラ駆動

図 2-4　モーターの駆動回路

2.2　フォトインタラプタ

モーターを制御するときに、原点の位置決めが必要になります。ここでは、光センサでモーターに取り付けられた円板の穴位置を検出し、そこを原点に決めています。

図 2-5 はフォトインタラプタ（光センサ）の概要です。現物はモーター基板上で円板を挟んでいる黒色の部品です。（a）は、その構造（断面）です。LED で光（赤外線）発し、対面するフォトトランジスタで光の透過 / 遮断を検知し、円板の穴位置（原点）を見つけます。（b）は基本回路です。常時赤外線 LED は発光しています。フォトトランジスタはベース電極の代わりに、接合部の受光量でエミッタ電流 Ie を制御します。受光時は Ie が流れ、出力（エミッタ電圧 Ve）が H レベルになります。遮光時は Ie が流れないため、出力が L レベルになります。



（a）断面図　　（b）基本回路

図 2-5　フォトセンサの構造・原理

3. 接続図の概要

図 3-1 はこのボードの接続図の概要を示します。VP1 のモーター駆動信号が、ユニポーラ駆動 IC（TD62083AF）とバイポーラ駆動 IC（TA7774FG）の両方に分岐入力されます。モーター基板の差し換えで 2 つの駆動 IC の一方の出力だけがモーターに接続されます。TD62083AF（トランジスタアレイ）は、トランジスタが 8 素子内蔵されているので、余裕を持たせて 2 素子を並列接続しています。

この図では省略されていますが、各信号の H/L レベルをモニターするステータス LED を設置しています。

接続図の詳細は MT-E506 取扱説明書（http://www.sunhayato.co.jp/products/item_data/MT-E506_Manual_SG097005.pdf）をご参照ください。

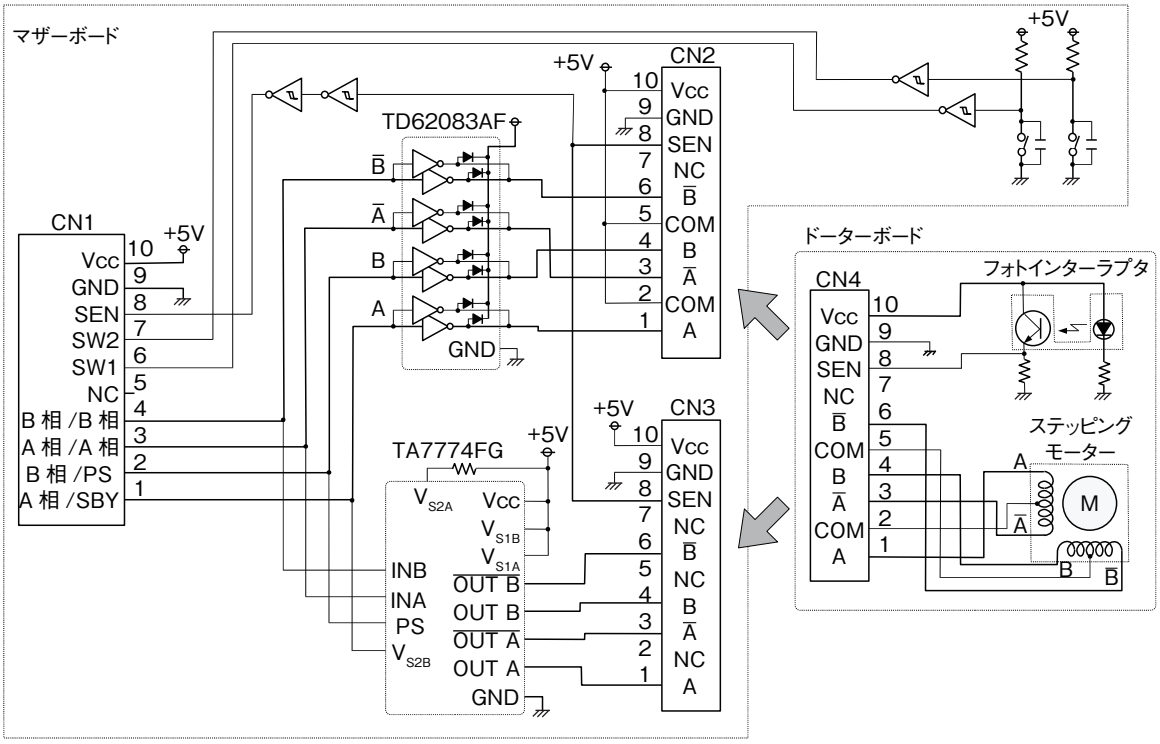


図 3-1　接続図の概要

4. 基本実習

このボードは 2 相のユニポーラ型モーターを搭載しています。最初に、そのままの結線でユニポーラ駆動の実習をします。次にその結線を変えてバイポーラ駆動の実習をします。マザーボードからモーター基板を抜いて、180 度まわして再びそれを差し込むことで、ユニポーラ駆動とバイポーラ駆動を切り替えます。このとき、マザーボード面にモーターの駆動方式を示す "Unipolar" または "Bipolar" の文字列が視認できます。実習の指示に沿って適宜切り替えてください。マイコンボードとステッピングモーターボードの接続は駆動方式にかかわらず共通です。図 4-1 のとおり、付属の 10P ケーブルにより MT-R300 の VPT と MT-E506 の VP1 を接続します。

基本プログラムを作成した後、それぞれの制御関数を作成し、それらをまとめたライブラリーを作成します。

4.1 ユニポーラ駆動

モーター基板を差し替えて、ユニポーラ駆動にしてください。ここでは、"X:¥xxxx¥workspace" にワークスペース、プロジェクト（stpm_u）を新規作成します。ハードウエアのセットアップ関数とメイン関数は自動生成しません。手動で "main.c" ファイル（空白）を新規作成し、プロジェクトに追加します。既に作成済みの "X:¥xxxx¥workspace¥@library" フォルダー内の "mt_r300.c" もプロジェクトに追加します。

種々のヘッダーファイルをインクルードするために次のとおり相対パスを指定します。

(1) [ビルド]、[H8S、H8/300 Standerd TOOLchain] を選択し、"H8S、H8/300 Standerd TOOLchain" ウインドウに切り替えます。

(2) [標準ライブラリー] タグ、[カテゴリ] で " ソース "、[オプション項目] で " インクルードファイルディレクトリ "（デフォルト値）を確認する。

(3)［追加］ボタンを押下して、［相対パス］で "Project directory" を選択し、［OK］を押下する。

(4) 同様に [追加] ボタンを押下して、[相対パス] で "Costom directory" を選択し、ディレクトリーに相対パスで "..¥..¥..¥@library" と入力し、[OK] を押下する。

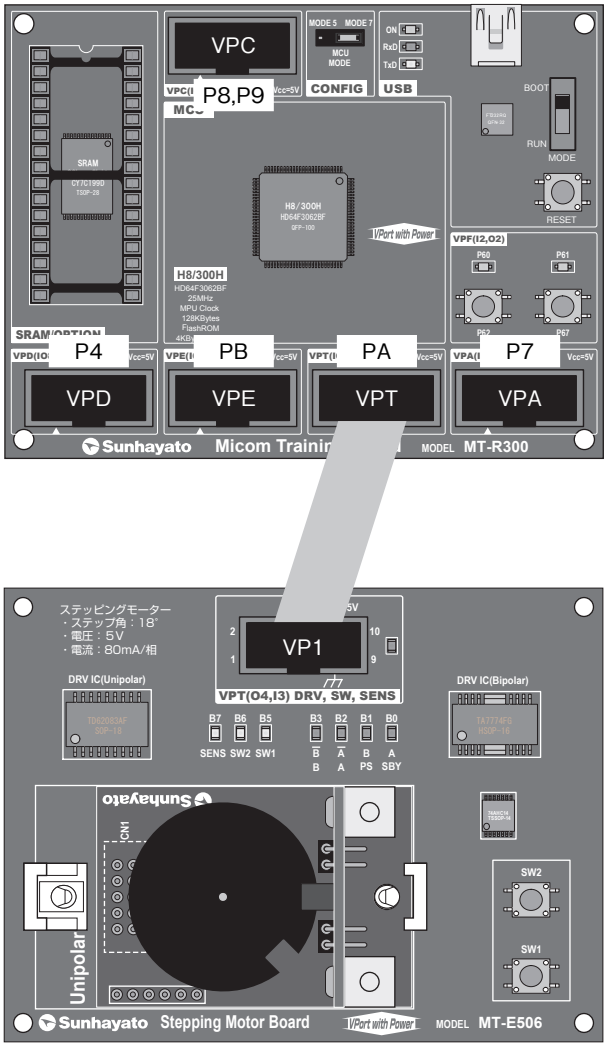


図 4-1　基板間の接続

実習 4.1.1a ユニポーラー駆動の 1 相励磁でモーターを左（右）に回転する

[解説]

図 2-3 のユニポーラ駆動の 1 相励磁のように励磁すると左回転します。コイル A、B、$\overline{A}$、$\overline{B}$ が、ビット 0、1、2、3 に対応しているため、0x01 → 0x02 → 0x04 → 0x08 →の順に出力します。モニター LED（b3 ～ b0）の 1 つが点灯し、左にローテーションします。逆順に出力すると右回転します。ウエイト時間により回転速度が変わります。

[解答]

リスト 4.1.1a にプログラム例を示します。

リスト 4.1.1a　1 相励磁で左（右）に回転する

```
/** main.c  ********************************************************************************/
#include      <mt_r300.h>                  // マイコンボード用ライブラリー
unsigned char rot_left( unsigned char d)   // 下位 4 ビットを左回転 ( 上位 4 ビットは入力モード )
{
    return (d & 0x08) >> 3 | ( d & 0x07) << 1;
}
unsigned char rot_rigt( unsigned char d)   // 下位 4 ビットを右回転 ( 上位 4 ビットは入力モード )
{
    return (d & 0x01) << 3 | ( d & 0x0e) >> 1;
}
void main (void)
{
    unsigned char d = 0x01;                // 1 相励磁
    setup_VPT(0x0f);  // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相 )
    out_VPT(d);
    wait_ms(200);
    while(1){
       d = rot_left(d);                    // 左回転
//     d = rot_right(d);                   // 右回転
       out_VPT(d);
       wait_ms(200);
    }
}
```

実習 4.1.1b ユニポーラー駆動の 1 相励磁でモーターを左（右）に回転する（引数を使わない方法）

[解説]

モーター制御関数で引数を使わない方法でプログラムを作成します。現状の励磁パターンを取り込み、それを基に次の励磁パターンを演算します。

[解答]

リスト 4.1.1b にプログラム例を示します。

リスト 4.1.1b　1 相励磁で左（右）に回転する（引数を使わない）

```
/** main.c  ********************************************************************************/
#include      <mt_r300.h>                  // マイコンボード用ライブラリー
void rot_left(void)                        // 下位 4 ビットを左回転 ( 上位 4 ビットは入力モード )
{
    unsigned char d;
    d = in_VPT() & 0x0f;                   // 現状を取り込む
    out_VPT((d & 0x08) >> 3 | ( d & 0x07) << 1);
```

```
}
void rot_right(void)                // 下位 4 ビットを右回転 ( 上位 4 ビットは入力モード )
{
    unsigned char d;
    d = in_VPT() & 0x0f;            // 現状を取り込む
    out_VPT((d & 0x01) << 3 | ( d & 0x0e) >> 1);
}
void main (void)
{
    setup_VPT(0x0f);  // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相 )
    out_VPT(0x01);                  // 1 相励磁
    wait_ms(200);
    while(1){
       rot_left();                  // 左回転
//     rot_right();                 // 右回転
       wait_ms(200);
    }
}
```

実習 4.1.2a ユニポーラー駆動の 2 相励磁でモーターを左（右）に回転する

[解説]

図 2-3 のユニポーラ駆動の 2 相励磁のように励磁すると左回転します。コイル A、B、$\overline{A}$、$\overline{B}$ が、ビット 0、1、2、3 に対応しているため、0x03 → 0x06 → 0x0c → 0x09 →の順に出力します。モニター LED(b3 ～ b0)の 2 つが点灯し、左にローテーションします。逆順に出力すると右回転します。ウエイト時間により回転速度が変わります。

[解答]

リスト 4.1.2a にプログラム例を示します。

リスト 4.1.2a　2 相励磁で左（右）に回転する

```
/** main.c  ********************************************************************************/
#include    <mt_r300.h>                         // マイコンボード用ライブラリー
unsigned char rot_left( unsigned char d)        // 下位 4 ビットを左回転 ( 上位 4 ビットは入力モード )
{
    return (d & 0x08) >> 3 | ( d & 0x07) << 1;
}
unsigned char rot_rigt( unsigned char d)        // 下位 4 ビットを右回転 ( 上位 4 ビットは入力モード )
{
    return (d & 0x01) << 3 | ( d & 0x0e) >> 1;
}
void main (void)
{
    unsigned char d = 0x03;                     // 2 相励磁
    setup_VPT(0x0f);  // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相 )
    out_VPT(d);
```

```
    wait_ms(200);
    while(1){
        d = rot_left(d);            // 左回転
//      d = rot_right(d);           // 右回転
        out_VPT(d);
        wait_ms(200);
    }
}
```

実習 4.1.2b ユニポーラー駆動の 2 相励磁でモーターを左（右）に回転する（引数を使わない方法）

[解説]

モーター制御関数で引数を使わない方法でプログラムを作成します。現状の励磁パターンを取り込み、それを基に次の励磁パターンを演算します。

[解答]

リスト 4.1.2b にプログラム例を示します。

リスト 4.1.2b　2 相励磁で左（右）に回転する（引数を使わない）

```
/** main.c  ********************************************************************************/
#include    <mt_r300.h>                     // マイコンボード用ライブラリー
void rot_left(void)                         // 下位 4 ビットを左回転（上位 4 ビットは入力モード）
{
    unsigned char d;
    d = in_VPT() & 0x0f;                    // 現状を取り込む
    out_VPT((d & 0x08) >> 3 | ( d & 0x07) << 1);
}
void rot_right(void)                        // 下位 4 ビットを右回転（上位 4 ビットは入力モード）
{
    unsigned char d;
    d = in_VPT() & 0x0f;                    // 現状を取り込む
    out_VPT((d & 0x01) << 3 | ( d & 0x0e) >> 1);
}
void main (void)
{
    setup_VPT(0x0f);  // 初期設定（b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相）
    out_VPT(0x03);                          // 2 相励磁
    wait_ms(200);
    while(1){
        rot_left();                         // 左回転
//      rot_right();                        // 右回転
        wait_ms(200);
    }
}
```

実習 4.1.3a ユニポーラー駆動の 1-2 相励磁でモーターを左（右）に回転する

[解説]

図 2-3 のユニポーラ駆動の 1-2 相励磁のように励磁すると左回転します。コイル A、B、$\overline{A}$、$\overline{B}$ が、ビット 0、1、2、3 に対応しているため、0x01 → 0x03 → 0x02 → 0x06 → 0x04 → 0x0c → 0x08 → 0x09 →の順に出力します。この励磁パターンはあらかじめ配列に格納しておき、それを順番に取り出します。モニター LED（b3 ～ b0）の 1 つ、または 2 つが交互に点灯し、左にローテーションします。逆順に出力すると右回転します。ウエイト時間により回転速度が変わります。

[解答]

リスト 4.1.3a にプログラム例を示します。

リスト 4.1.3a　1-2 相励磁で左（右）に回転する

```
** main.c  ***************************************************************************/
#include    <mt_r300.h>             // マイコンボード用ライブラリー
void main (void)
{
    unsigned char ptn_12[] = {0x01,0x03,0x02,0x06,0x04,0x0c,0x08,0x09}; // 1-2 相励磁パターン
    int i = 0;
    setup_VPT(0x0f);  // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相 )
    out_VPT(ptn_12[i]);
    wait_ms(200);
    while(1){
        i = (i + 1) & 0x07;         // 左回転演算
//      i = (i - 1) & 0x07;         // 右回転演算
        out_VPT(ptn_12[i]);
        wait_ms(200);
    }
}
```

実習 4.1.3b ユニポーラー駆動の 1-2 相励磁でモーターを左(右)に回転する(配列を検索する方法)

[解説]

モーター制御関数では現状の励磁パターンを取り込み、それが配列の何番目か検索し、その次の励磁パターンを求めます。引数が不要になります。

[解答]

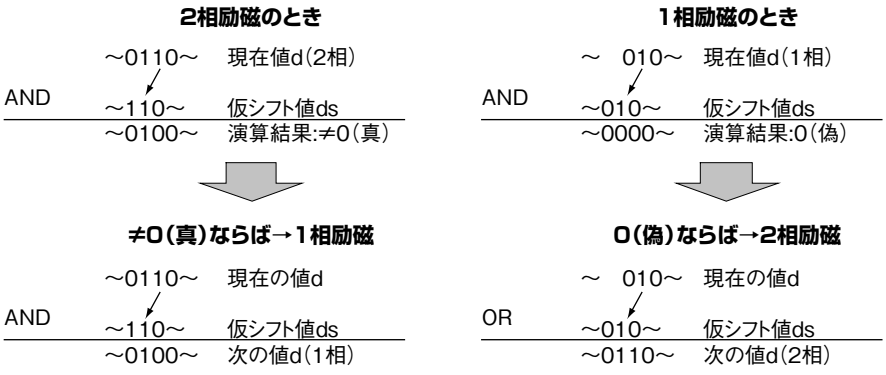
リスト 4.1.3b にプログラム例を示します。

リスト 4.1.3b　1-2 相励磁で左（右）に回転する（配列検索）

```
** main.c  ***************************************************************************/
#include    <mt_r300.h>             // マイコンボード用ライブラリー
void rot_left(void)                 // 下位 4 ビットを左回転 ( 上位 4 ビットは入力モード )
{
    unsigned char d;
```

```
    unsigned char ptn_12[] = {0x01,0x03,0x02,0x06,0x04,0x0c,0x08,0x09}; // 1-2 相励磁パターン
    int i = 0;
    d = in_VPT() & 0x0f;            // 現状を取り込む
    while( d != ptn_12[i++] );      // 励磁パターンの検索（i++ を求める）
    out_VPT(ptn_12[i & 0x07]);
}
void rot_right(void)                // 下位 4 ビットを右回転（上位 4 ビットは入力モード）
{
    unsigned char d;
    unsigned char ptn_12[] = {0x01,0x03,0x02,0x06,0x04,0x0c,0x08,0x09}; // 1-2 相励磁パターン
    int i = 7;
    d = in_VPT() & 0x0f;            // 現状を取り込む
    while( d != ptn_12[i--] );      // 励磁パターンの検索（i-- を求める）
    out_VPT(ptn_12[i & 0x07]);
}
void main (void)
{
    setup_VPT(0x0f);  // 初期設定（b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相）
    out_VPT(0x01);                  // 初期値
    wait_ms(200);
    while(1){
        rot_left();                 // 左回転
//      rot_right();                // 右回転
        wait_ms(200);
    }
}
```

実習 4.1.3c ユニポーラー駆動の 1-2 相励磁でモーターを左（右）に回転する（演算で求める方法）



（プログラム例）

```
d = in_VPT() & 0x0f;                        // 現在値
ds = (d & 0x08) >> 3 | ( d & 0x07) << 1;    // 仮シフト値
if(d & ds)                                  // 真(2相励磁)のとき
    d = d & ds;                             // 次は1相励磁に
else                                        // 偽(1相励磁)のとき
    d = d | ds;                             // 次は2相励磁に
```

図 4-2　1-2 相励磁パターンの演算

[解説]

モーター制御関数では現状の励磁パターンを取り込み、演算によりその次の励磁パターンを求めます。

図 4-2 は励磁パターンの演算方法の一例を示します。現在値と仮シフト値のビット積（AND）は、2 相励磁のとき真（≠ 0）、1 相励磁のとき偽（= 0）となります。これを踏まえ、2 相励磁のときは両者のビット積（AND）を求め 1 相励磁にし、1 相励磁のときは両者のビット和（OR）を求め 2 相励磁にします。

[解答]

リスト 4.1.3c にプログラム例を示します。

リスト 4.1.3c　1-2 相励磁で左（右）に回転する（演算で求める）

```
** main.c  ********************************************************************************/
#include    <mt_r300.h>             // マイコンボード用ライブラリー
void rot_left(void)                 // 下位 4 ビットを左回転 ( 上位 4 ビットは入力モード )
{
    unsigned char d, ds;
    d = in_VPT() & 0x0f;                            // 現状を取り込む
    ds = (d & 0x08) >> 3 | ( d & 0x07) << 1;        // 左回転
    if( d & ds )                                    // 2 相励磁のとき
        d = d & ds;                                 // 次は 1 相励磁
    else                                            // 1 相励磁のとき
        d = d | ds;                                 // 次は 2 相励磁
    out_VPT(d);
}
void rot_right(void)                                // 下位 4 ビットを右回転 ( 上位 4 ビットは入力モード )
{
    unsigned char d, ds;
    d = in_VPT() & 0x0f;                            // 現状を取り込む
    ds = (d & 0x01) << 3 | ( d & 0x0e) >> 1;        // 右回転
    if( d & ds )                                    // 2 相励磁のとき
        d = d & ds;                                 // 次は 1 相励磁
    else                                            // 1 相励磁のとき
        d = d | ds;                                 // 次は 2 相励磁
    out_VPT(d);
}
void main (void)
{
    unsigned char d;
    setup_VPT(0x0f);  // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相 )
    out_VPT(0x01);                                  // 初期値
    wait_ms(200);
    while(1){
        rot_left();                                 // 左回転
//      rot_right();                                // 右回転
        wait_ms(200);
    }
```

```
}
```

4.2　ユニポーラ駆動関数のライブラリー作成

4.1 で作成したモーター駆動関数をライブラリーにまとめます。ここでは "X:¥xxxx¥workspace" にワークスペース、プロジェクト（stpm_u_lib）を新規作成し、4.1 と同様に種々の設定をします。

モーター駆動関数をまとめるライブラリー（"mt_e506.c"、"mt_e506.h"）を新規作成します。まず " ファイル "、" 新規作成 " で 2 ファイル（空白の Document1、Document2 など）を作成します。" 名前を付けて保存 " で空白のそれぞれを、"mt_e506.c"，"mt_e506.h" という名前でフォルダー "X:¥xxxx¥workspace¥@library" に保存します。このフォルダーに "mt_r300.c"、"mt_r300.h" などと同様に、2 つのファイルが追加格納されていることを確認します。

続いて "mt_r300.c" と同様に "mt_e506.c" もプロジェクトに追加します。

実習 4.2.1　ユニポーラー駆動の 1 相励磁でモーターを左（右）に回転する（ライブラリー使用）

[解説]

ライブラリー "mt_e506.c" に実習 4.1.1 のモーター駆動関数を移植します。合わせて "mt_e506.h" にその関数のプロトタイプ宣言を移植します。

[解答]

リスト 4.2.1 にプログラム例を示します。

リスト 4.2.1　1 相励磁で左（右）に回転する（ライブラリー使用）

```
/** main.c  ********************************************************************************/
#include    <mt_r300.h>             // マイコンボード用ライブラリー
#include    <mt_e506.h>             // ステッピングモータボード用ライブラリー

void main (void)
{
    initialize_stpm_u1();           // モータ初期化
    wait_ms(100);
    while(1){
        left_stpm_u1();             // モータ左回転
//      right_stpm_u1();            // モータ右回転
        wait_ms(100);
    }
}
```

```
/** mt_e506.c ( 新規作成 ) ********************************************************************/
#include    <iodefine.h>
#include    <mt_r300.h>             // マイコンボード用ライブラリー
/*****  ユニポーラ駆動 ( 1 相励磁 )  *****/
void initialize_stpm_u1( void )     // 初期設定する関数
{
    setup_VPT(0x0f); // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相 )
    out_VPT(0x01);                          // 初期位置
```

```
}
void left_stpm_u1( void )    // 左に 1 ステップ（18 度）回転する関数
{
    unsigned char d;
    d = in_VPT() & 0x0f;                    // 現状データを取り込む
    out_VPT((d & 0x08) >> 3 | ( d & 0x07) << 1);
}
void right_stpm_u1( void )   // 右に 1 ステップ（18 度）回転する関数
{
    unsigned char d;
    d = in_VPT() & 0x0f;                    // 現状データを取り込む
    out_VPT((d & 0x01) << 3 | ( d & 0x0e) >> 1);
}
```

```
/** mt_e506.h（新規作成）*************************************************************/
/*****  ユニポーラ駆動（1 相励磁）  *****/
void initialize_stpm_u1( void );
void left_stpm_u1( void );
void right_stpm_u1( void );
```

実習 4.2.2　ユニポーラー駆動の 2 相励磁でモーターを左（右）に回転する（ライブラリー使用）

[解説]

ライブラリー "mt_e506.c" に実習 4.1.2 のモーター駆動関数を追加移植します。合わせて "mt_e506.h" にその関数のプロトタイプ宣言を追加移植します。

[解答]

リスト 4.2.2 にプログラム例を示します。

リスト 4.2.2　2 相励磁で左（右）に回転する（ライブラリー使用）

```
/** main.c  ***************************************************************************/
#include    <mt_r300.h>             // マイコンボード用ライブラリー
#include    <mt_e506.h>             // ステッピングモータボード用ライブラリー
void main (void)
{
    initialize_stpm_u2();           // モータ初期化
    wait_ms(100);
    while(1){
        left_stpm_u2();             // モータ左回転
//      right_stpm_u2();            // モータ右回転
        wait_ms(100);
    }
}
```

```
/** mt_e506.c ( 以下を追加します ) ******************************************************/
/*****  ユニポーラ駆動 (2 相励磁 )  *****/
void initialize_stpm_u2( void )            // 初期設定する関数
{
    setup_VPT(0x0f);  // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相 )
    out_VPT(0x03);                         // 初期位置
}
void left_stpm_u2( void )    // 左に 1 ステップに 1 ステップ (18 度 ) 回転する関数
{
    unsigned char d;
    d = in_VPT() & 0x0f;                   // 現状データを取り込む
    out_VPT((d & 0x08) >> 3 | ( d & 0x07) << 1);
}
void right_stpm_u2( void )  // 右に 1 ステップ (18 度 ) 回転する関数
{
    unsigned char d;
    d = in_VPT() & 0x0f;                   // 現状データを取り込む
    out_VPT((d & 0x01) << 3 | ( d & 0x0e) >> 1);
}
```

```
/** mt_e506.h ( 以下を追加します ) ******************************************************/
/*****  ユニポーラ駆動 (2 相励磁 )  *****/
void initialize_stpm_u2( void );
void left_stpm_u2( void );
void right_stpm_u2( void );
```

実習 4.2.3　ユニポーラー駆動の 1-2 相励磁でモーターを左 (右) に回転する (ライブラリー使用)

[解説]

ライブラリー "mt_e506.c" に実習 4.1.3 のモーター駆動関数を追加移植します。合わせて "mt_e506.h" にその関数のプロトタイプ宣言を追加移植します。

[解答]

リスト 4.2.3 にプログラム例を示します。

リスト 4.2.3　1-2 相励磁で左 (右) に回転する (ライブラリー使用)

```
/** main.c  ***************************************************************************/
#include    <mt_r300.h>              // マイコンボード用ライブラリー
#include    <mt_e506.h>              // ステッピングモータボード用ライブラリー
void main (void)
{
    initialize_stpm_u12();           // モータ初期化
    wait_ms(100);
    while(1){
        left_stpm_u12();             // モータ左回転
```

```
//      right_stpm_u12();           // モータ右回転
        wait_ms(100);
    }
}
```

```
/** mt_e506.c ( 以下を追加します ) ********************************************************/
/*****  ユニポーラ駆動 (1-2 相励磁 )  *****/
void initialize_stpm_u12( void )          // 初期設定する関数
{
    setup_VPT(0x0f);  // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B~ 相 ,b2:A~ 相 ,b1:B 相 ,b0:A 相 )
    out_VPT(0x01);                        // 初期位置
}
void left_stpm_u12( void )  // 左に 1 ステップ (9 度 ) 回転する関数
{
    unsigned char d, ds;
    d = in_VPT() & 0x0f;                  // 現状データを取り込む
    ds = (d & 0x08) >> 3 | ( d & 0x07) << 1;
    out_VPT((d & ds) ? (d & ds) : (d | ds) );
}
void right_stpm_u12( void ) // 右に 1 ステップ (9 度 ) 回転する関数
{
    unsigned char d, ds;
    d = in_VPT() & 0x0f;                  // 現状データを取り込む
    ds = (d & 0x01) << 3 | ( d & 0x0e) >> 1;
    out_VPT((d & ds) ? (d & ds) : (d | ds) );
}
```

```
/** mt_e506.h ( 以下を追加します ) ********************************************************/
/*****  ユニポーラ駆動 (1-2 相励磁 )  *****/
void initialize_stpm_u12( void );
void left_stpm_u12( void );
void right_stpm_u12( void );
```

実習 4.2.4 ユニポーラー駆動の 1-2 相励磁で原点復帰し、SW2 で左（右）回転し、SW1 で停止する

[解説]

ライブラリー "mt_e506.c" にセンサ、スイッチの信号を取り込む関数を追加します。合わせて "mt_e506.h" にその関数のプロトタイプ宣言を追加します。SW1、SW2 を押し、フォトセンサ部に円板の穴を合わせ、モニタ LED（b7 ～ b5）の点灯を確認します。

[解答]

リスト 4.2.4 にプログラム例を示します。

リスト 4.2.4　1-2 相励磁で原点復帰し、SW2 で左（右）回転し、SW1 で停止する

```
/** main.c  ********************************************************************************/
#include    <mt_r300.h>                     // マイコンボード用ライブラリー
#include    <mt_e506.h>                     // ステッピングモータボード用ライブラリー
void main (void)
{
    int rot_flag;
    initialize_stpm_u12();                  // モータ初期化
    wait_ms(100);
    while(1){
        while( !get_sensor() ){             // 原点復帰（フォトセンサ 1 になるまで待つ）
            right_stpm_u12();               // モータ右回転
            wait_ms(200);
        }
        rot_flag = 0;
        while(1){
            if(get_sw2())
                rot_flag = 1;
            if(get_sw1())
                rot_flag = 0;
            if(rot_flag)
                left_stpm_u12();            // モータ左回転
//              right_stpm_u12();           // モータ右回転
                wait_ms(100);
        }
    }
}
```

```
/** mt_e506.c（以下を追加します）*************************************************/
/*****  他の関数（センサ，スイッチの入力）*****/
int get_sensor()                            // フォトセンサの信号を取り込む
{
    return in_VPT() >> 7;                   // スリット検出 :1/ スリットなし :0
}
int get_sw2()                               // SW2 の信号を取り込む
{
    return (in_VPT() & 0x40) >> 6;          // SW ON:1/OFF:0
}
int get_sw1()                               // SW1 の信号を取り込む
{
    return (in_VPT() & 0x20) >> 5;          // SW ON:1/OFF:0
}
```

```
/** mt_e506.h（以下を追加します）*************************************************/
/*****  他の関数（センサ，スイッチの入力）  *****/
```

```
int get_sensor( void );
int get_sw2( void );
int get_sw1( void );
```

4.3　バイポーラ駆動

モーター基板を差し替えて、バイポーラ駆動に切り替えてください。ここでは、"X:\xxxx\workspace" にワークスペース、プロジェクト（stpm_b）を新規作成します。4.1 と同様に種々の設定をします。

実習 4.3.1a バイポーラー駆動（2 相励磁）で左（右）に回転する

[解説]

図 2-3 のバイポーラー駆動（2 相励磁）のように励磁すると左回転します。コイル A、B に、0x00 → 0x01 → 0x03 → 0x02 →の順に出力します。実際には INA、INB が、ビット 2、3 に対応しているため、その値を左に 2 ビットシフトして出力します。モニター LED（b3 ～ b2）の 2 つが指示どおりに点灯します。逆順に出力すると右回転します。ウエイト時間により回転速度が変わります。

[解答]

リスト 4.3.1a にプログラム例を示します

リスト 4.3.1a　バイポーラー駆動 (2 相励磁) で左（右）に回転する

```
/** main.c  ********************************************************************************/
#include    <mt_r300.h>                             // マイコンボード用ライブラリー
unsigned char ptn_ab[] = {0x00,0x01,0x03,0x02};   // A,B 相：00 01 11 10
void main (void)
{
    int    i = 0;
    setup_VPT(0x0f);   // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B 相 ,b2:A 相 ,b1:PS,b0:STB)
    out_VPT(0x03);                                  // b3:0(B 相 ), b2:0(A 相 ), b1:0(PS), b0:1(STB)
    wait_ms(200);
    while(1){
        i = (i+1) & 0x03;                           // 左回転
//      i = (i-1) & 0x03;                           // 右回転
        out_VPT( (ptn_ab[i] << 2)| 0x03 );          // b3:B 相, b2:A 相, b1:1(PS), b0:1(STB)
        wait_ms(200);
    }
}
```

実習 4.3.1b バイポーラー駆動（2 相励磁）で左（右）に回転する（配列を検索し次の値を求める）

[解説]

モーター制御関数で引数を使わない方法でプログラムを作成します。現状の励磁パターンを取り込み、それが配列の何番目か検索し、その次の励磁パターンを求めます。

[解答]

リスト 4.3.1b にプログラム例を示します。

リスト 4.3.1b　バイポーラー駆動（2 相励磁）で左（右）に回転する（配列検索）

```
/** main.c  ********************************************************************************/
#include    <mt_r300.h>                                     // マイコンボード用ライブラリー
void rot_left(void)                                         // 左回転（18 度 ）
{
    unsigned char ptn_ab[] = {0x00,0x01,0x03,0x02};         // A,B 相：00 01 11 10
    unsigned char d, ab;
    int i;
    d = in_VPT();
    ab = (d & 0x0c) >> 2;                                   // 現状データを取り込む
    i = 0;
    while(ab != ptn_ab[i++]);                               // 励磁パターン検索（+1 で終了 ）
    out_VPT( ptn_ab[i & 0x03] << 2 | d & 0x03);             // i で左回転
}
void rot_right(void)                                        // 右回転（18 度 ）
{
    unsigned char ptn_ab[] = {0x00,0x01,0x03,0x02};         // A,B 相：00 01 11 10
    unsigned char d, ab;
    int i;
    d = in_VPT();
    ab = (d & 0x0c) >> 2;                                   // 現状データを取り込む
    i = 0;
    while(ab != ptn_ab[i++]);                               // 励磁パターン検索（+1 で終了 ）
    out_VPT( ptn_ab[i-2 & 0x03] << 2 | d & 0x03);           // (i-2) で右回転
}
void main (void)
{
    setup_VPT(0x0f);  // 初期設定（b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B 相 ,b2:A 相 ,b1:PS,b0:STB)
    out_VPT(0x03);    // b3:0(B 相 ), b2:0(A 相 ), b1:1( パワーセーブ ), b0:1( 動作 )
    wait_ms(200);
    while(1){
        rot_left();
//      rot_right();
        wait_ms(200);
    }
}
```

実習 4.3.1c　バイポーラー駆動（2 相励磁）で左（右）に回転する（演算で求める方法）

[解説]

モーター制御関数では現状の励磁パターンを取り込み、演算によりその次の励磁パターンを求めます。

励磁パターンの演算方法の一例を示します。

励磁パターン配列　unsigned char ptn_ab[] = {0x00,0x01,0x03,0x02}; は、
10 進数で書くと、 unsigned char ptn_ab[] = { 0, 1, 3, 2}; です。
これは、2 番目と 3 番目の中身がそれぞれ 3、2 と逆になっています。
そこで現在の値を読み出した後、2 と 3 を入れ替えてから検索し、次の値を求めた後、再度 2 と 3 を入れ替えて出力する方法を試みましょう。

[解答]

リスト 4.3.1c にプログラム例を示します

リスト 4.3.1c　バイポーラー駆動（2 相励磁）で左（右）に回転する（演算で求める）

```
/** main.c  ********************************************************************************/
#include    <mt_r300.h>             // マイコンボード用ライブラリー
void rot_left(void)                 // 左回転
{
    unsigned char d, ab;
    d = in_VPT();
    ab = (d & 0x0c) >> 2;           // 現状データを取り込む
    if(ab == 3)                     // 2,3 を逆にする
        ab = 2;
    else if(ab == 2)
             ab = 3;
    ab = (ab + 1) & 0x03;           // 左回転
    if(ab == 3)                     // 2,3 を逆にする
        ab = 2;
    else if(ab == 2)
             ab = 3;
    out_VPT( ab << 2 | d & 0x03);
}
void rot_right(void)                // 右回転
{
    unsigned char d, ab;
    d = in_VPT();
    ab = (d & 0x0c) >> 2;           // 現状データを取り込む
    if(ab == 3)                     // 2,3 を逆にする
        ab = 2;
    else if(ab == 2)
             ab = 3;
    ab = (ab - 1) & 0x03;           // 右回転
    if(ab == 3)                     // 2,3 を逆にする
        ab = 2;
    else if(ab == 2)
             ab = 3;
    out_VPT( ab << 2 | d & 0x03);
}
void main (void)
```

```
{
    setup_VPT(0x0f);    // 初期設定（b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B 相 ,b2:A 相 ,b1:PS,b0:STB)
    out_VPT(0x03);      // b3:0(B 相 ), b2:0(A 相 ), b1:1( パワーセーブ ), b0:1( 動作 )
    wait_ms(100);
    while(1){
        rot_left();         // 左回転
//      rot_right();        // 右回転
        wait_ms(100);
    }
}
```

実習 4.3.2　バイポーラー駆動（2 相励磁）で左（右）に回転する（ITU0 のトグル出力機能を使う方法）

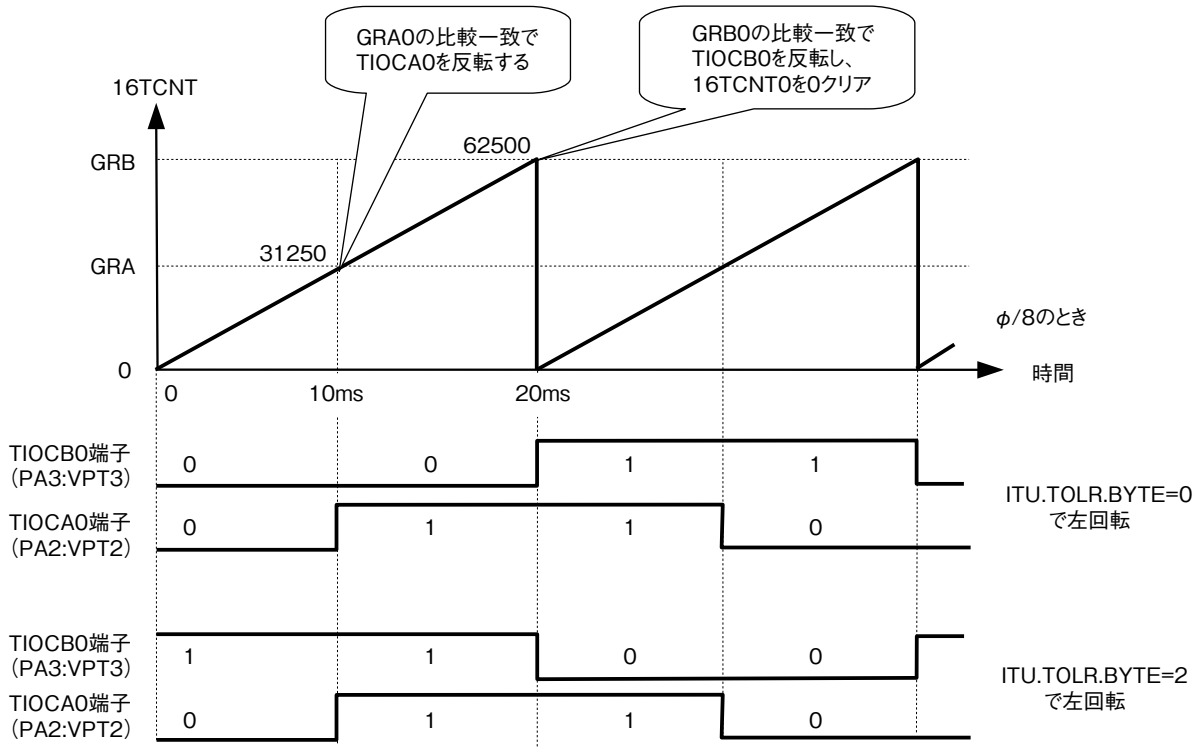


図 4-3　ITU0 の 2 相トグル出力機能

[解説]

図 4-3 は ITU0 の 2 相トグル出力機能の概要を示します。まず内部クロック φ（25MHz）を 1/8 に分周し、3.125MHz のクロックを作ります。それを 16TCNT0 でカウントし、GRA0 値（31250）と比較します。10m 秒後に 16TCNT0 と GRA0 が一致し、TIOCA0 信号が反転します。更に 10m 秒後（最初から 20m 秒後）に 16TCNT0 と GRB0 が一致し、TIOCB0 信号が反転し、16TCNT0 を 0 クリアします。これを繰り返します。

このように ITU のレジスタに必要な設定をするだけで、TIOCB0（PA3:VPT3）、TIOCA0（PA2:VPT2）の 2 端子にトグル（0、1 を交互に繰り返す）信号を出力します。

図 4-4 は ITU0 のレジスタの設定方法を示します。

タイマ I/O コントロールレジスタ TIOR：GRB0 の比較一致で 16CNT0 クリア、内部クロック , φ /8

タイマ I/O コントロールレジスタ TIOR：16TCNT0 と GRB0、GRA0 の比較一致でそれぞれトグル出力

ジェネラルレジスタ GRA、GRB：1 ステップの時間を 10m 秒にするときは GRB に 20m 秒、GRA にその半分の

値を設定します。

図 4-4　ITU0 のレジスタの設定

タイマアウトプットレベルセットレジスタ（TOLR）の初期値で回転の向きが決まります（TIOB0=0、TIOA0=0 で左回転、TIOB0=1、TIOA0=0 で右回転）。

[解答]

リスト 4.3.2 にプログラム例を示します

リスト 4.3.2　バイポーラー駆動で左（右）に回転する（ITU0 のトグル出力機能）

```
/** main.c  ********************************************************************************/
#include    <iodefine.h>
#include    <mt_r300.h>       // マイコンボード用ライブラリー
void main (void)
```

```
{
    long t;
    setup_VPT(0x0f);  // 初期設定 (b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B 相 ,b2:A 相 ,b1:PS,b0:STB)
    out_VPT(0x01);    // b3:0(B 相 ),b2:0(A 相 ),b1:0( ノーマルパワー ),b0:1( 動作 ):( 高速回転対応 )
    wait_ms(100);
    ITU0.TCR.BYTE = 0x43;           // TCNT0 と GRB0 の比較一致で TCNT0 クリア ,1/8 φ
    ITU0.TIOR.BIT.IOB = 3;          // TCNT0 と GRB0 の比較一致で TIOCB0 端子 (PA3:VPT3) のトグル出力
    ITU0.TIOR.BIT.IOA = 3;          // TCNT0 と GRA0 の比較一致で TIOCA0 端子 (PA2:VPT2) のトグル出力
    ITU.TOLR.BYTE = 0x00;           // 左方向 ( 初期値 :IOB0(0),IOA0(0))
//  ITU.TOLR.BYTE = 0x02;           // 左方向 ( 初期値 :IOB0(1),IOA0(0))
    wait_ms(1000);                  // 開始位置まで回転
    ITU0.TCNT = 0;                  // 16TCNT0 のクリア
    ITU0.GRB = 62500-1;             // 全周期 [25MHz/8=3.125MHz(0.32 μ秒 ),0.32 μ秒× 62500=20m 秒 ]
                                    // 21m 秒 ( 最大 ) ～ 10m 秒程度 ( これ未満では追随が難しい )
    ITU0.GRA = (ITU0.GRB+1)/2-1;    // 半周期 (1 ステップの時間 :10m 秒 )
    while(1){
        ITU.TSTR.BIT.STR0 = 1;      // TCNT0 のカウント動作 ( モータ回転 )
        wait_ms_itu2(400);          // 1 ステップ 10m 秒の時 :2 回転 (400m 秒 /10m 秒 /20 ステップ )
        ITU.TSTR.BIT.STR0 = 0;      // TCNT0 のカウント停止 ( モータ停止 )
        wait_ms_itu2(1000);
    }
}
```

全周期の最大値は約 21m 秒です。もっと遅くしたい場合は、適切な外部クロックを入力します。このモーターは、ステップ角が 18 度と大きいので 10m 秒未満に設定すると追随できなくなります（モーターが振動し回転しません)。ステップ角が 1.8 度程度と小さいモーターでは、1m 秒程度でも追随するでしょう。

4.4　バイポーラ駆動関数のライブラリー作成

4.2 で作成したライブラリー（"mt_e506.c", "mt_e506.h"）に 4.3 で作成したモーター駆動関数を追加移植します。ここでは､"X:¥xxxx¥workspace" にワークスペース､プロジェクト (stpm_b_lib) を新規作成します。4.2 と同様に種々の設定をし、"mt_e506.c" をプロジェクトに追加します。

実習 4.4.1　バイポーラー駆動（2 相励磁）で左（右）に回転する（ライブラリー使用）

[解説]

ライブラリー "mt_e506.c" に実習 4.3.1 のモーター駆動関数を追加移植します。合わせて "mt_e506.h" にその関数のプロトタイプ宣言を追加移植します。バイポーラ駆動 (2 相励磁) モーターの初期設定関数では､引数 ps (1: パワーセーブ／ 0: ノーマルパワー)、stb（1: 動作／ 0: 停止）を設定します。

[解答]

リスト 4.4.1 にプログラム例を示します。

リスト 4.4.1　左（右）に回転する（ライブラリー使用）

```
/** main.c  ********************************************************************************/
```

```
#include    <mt_r300.h>         // マイコンボード用ライブラリー
#include    <mt_e506.h>         // ステッピングモータボード用ライブラリー
void main (void)
{
    initialize_stpm_b(1, 1);    // 初期設定(ps(1:パワーセーブ),stb(1:動作))
    while(1){
        left_stpm_b();          // 左回転
//      right_stpm_b();         // 右回転
        wait_ms(50);
    }
}
```

```
/** mt_e506.c (以下を追加します) ********************************************************/
/*****  バイポーラ駆動(2相励磁)  *****/
void initialize_stpm_b( int ps, int stb )  // 初期設定
{                                   // ps(1:パワーセーブ/0:ノーマルパワー),stb(1:動作/0:停止)
    setup_VPT(0x0f);  // 初期設定(b7:SEN,b6:SW2,b5:SW1,b4:NC,b3:B相,b2:A相,b1:ps,b0:stb)
    out_VPT( ps << 1 | stb );       // b3:0(B相), b2:0(A相), b1:ps, b0:stb
    wait_ms(100);
}
void left_stpm_b( void )            // 左に1ステップ(18度)回転する(0_1_3_2_0_の繰り返し)
{
    unsigned char d, ab;
    d = in_VPT() & 0x0f;
    ab = (d & 0x0c) >> 2;
    if(ab == 3)                     // 2,3を逆にする
        ab = 2;
    else if(ab == 2)
            ab = 3;
    ab = (ab + 1) & 0x03;           // 左回転
    if(ab == 3)                     // 2,3を逆にする
        ab = 2;
    else if(ab == 2)
            ab = 3;
    out_VPT( ab << 2 | d & 0x03);
}
void right_stpm_b( void )           // 右に1ステップ(18度)回転する(0_2_3_1_0の繰り返し)
{
    unsigned char d, ab;
    d = in_VPT() & 0x0f;
    ab = (d & 0x0c) >> 2;
    if(ab == 3)
        ab = 2;
    else if(ab == 2)                // 2,3を逆にする
            ab = 3;
```

```
    ab = (ab - 1) & 0x03;            // 右回転
    if(ab == 3)
        ab = 2;
    else if(ab == 2)                 // 2,3 を逆にする
             ab = 3;
    out_VPT( ab << 2 | d & 0x03);
}
```

```
/** mt_e506.h ( 以下を追加します ) ************************************************/
/*****  バイポーラ駆動 (2 相励磁 )  *****/
void initialize_stpm_b( int, int );
void left_stpm_b( void );
void right_stpm_b( void );
```

実習 4.4.2　原点復帰し、SW2 で左（右）に回転し、SW1 で停止する（ライブラリー使用）

[解答]

リスト 4.4.2 にプログラム例を示します。

リスト 4.4.2　原点復帰し、SW2 で左（右）に回転し、SW1 で停止する（ライブラリー使用）

```
/** main.c  ***************************************************************************/
#include    <mt_r300.h>              // マイコンボード用ライブラリー
#include    <mt_e506.h>              // ステッピングモータボード用ライブラリー

void main (void)
{
    int rot_flag;
    initialize_stpm_b(1, 1);         // 初期設定 (PS(1: パワーセーブ ),STB(1: 動作 ))
    while( !get_sensor() ){          // 原点復帰 ( フォトインタラプタが 1 になるまで繰り返す )
        right_stpm_b();              // 右回転
        wait_ms(100);
    }
    rot_flag = 0;                    // フラグ：停止
    while(1){
        if(get_sw2())
            rot_flag = 1;            // フラグ：回転
        if(get_sw1())
            rot_flag = 0;            // フラグ：停止
        if(rot_flag)
            left_stpm_b();           // 左回転
//          right_stpm_b();          // 右回転
            wait_ms(50);
    }
}
```

実習 4.4.3　原点復帰し、SW2 で左（右）に回転し、指定位置に達したら停止する（ITU0 を用いる方法）

[解説]

通常のウエイト関数はその内部で指定時間ループしています。従って、その間メイン関数は何もできません。そこで ITU の比較一致フラグとループカウンターを見て指定時間に達したか（真 / 偽）を返す関数を作成します。メイン関数はその関数を呼び出し、その返値を受けて指定時間に達したかを判断する処理を繰り返します。指定時間に達したときは比較一致フラグとループカウンターなどをクリアし、同じ処理を繰り返します。この間メイン関数は他の処理が可能です。

[解答]

リスト 4.4.3 にプログラム例を示します。

リスト 4.4.3　原点復帰し、SW2 で回転し、指定位置で停止する（ITU0 を用いる）

```
/** main.c  ********************************************************************************/
#include    <iodefine.h>
#include    <mt_r300.h>                 // マイコンボード用ライブラリー
#include    <mt_e506.h>                 // ステッピングモータボード用ライブラリー
unsigned char count_50ms(void)
{
    static unsigned int roop_cnt = 0;
    unsigned char end_flg = 0;
    if(ITU.TISRA.BIT.IMFA0 == 1){       // 比較一致フラグ == 1
        ITU.TISRA.BIT.IMFA0 = 0;        // ITU0 の比較一致フラグクリア
        roop_cnt++;
        if(roop_cnt >= 5){              // 10ms × 5=50ms
            roop_cnt = 0;
            end_flg = 1;
        }
    }
    return end_flg;
}
void main (void)
{
    unsigned int step_cnt, step;
    initialize_stpm_b(1, 1);            // 初期設定 (PS(1: パワーセーブ ),STB(1: 動作 ))
    while( !get_sensor() ){             // 原点復帰 ( フォトインタラプタが 1 になるまで繰り返す )
        right_stpm_b();                 // 右回転
        wait_ms(100);
    }
    ITU0.TCR.BYTE = 0x23;               // TCNT と GRA の比較一致で TCNT クリア ,1/8 φ
    ITU0.GRA = 31250 - 1;           // 周期 :[25MHz/8=3.125MHz(0.32 μ秒 ),0.32 μ秒× 31250=10m 秒 ]
    ITU0.TCNT = 0;                      // TCNT クリア
    ITU.TISRA.BIT.IMFA0 = 0;            // TCNT と GRA の比較一致フラグクリア
    ITU.TISRA.BIT.IMIEA0 = 0;           // ITU0 の割り込み禁止
```

```
    while(1){
        while( !get_sw2() );            // SW2 を押すまで待つ
        step_cnt = 0, step = 40;        // 2 回転 (40 ステップ /20)
        ITU.TSTR.BIT.STR0 = 1;          // TCNT のカウントスタート
        while(step_cnt < step){
            if(count_50ms()){           // 50m 秒に達したか？
                right_stpm_b();
                step_cnt++;
            }
        }
        ITU.TSTR.BIT.STR0 = 0;          // TCNT のカウント停止
    }
}
```

5. おわりに

この実習によりステッピングモーターを制御する基本プログラムが理解できたことと思います。モーター基板の差し替えにより、ユニポーラ駆動とバイポーラ駆動の違い、1 相、2 相、1-2 相の各励磁方式についても理解されたことと思います。各信号のレベル（H/L）の変化を視認するモニタ LED も効果的だったと思います。それらの制御プログラムをライブラリーにまとめましたので、今後はそれを活用して種々の制御プログラムを作成しましょう。

モーターは慣性モーメントのため、急な回転、停止はできません。徐々に回転速度を上げたり、下げたりする、加速－定速－減速の制御プログラムも作成できますが、難しくなります。LCD ボード MT-E502 と組み合わせることでモーターの回転位置、回転速度なども表示できます。これらの応用的な実習は別のテキストに譲ります。

マイコントレーニング基板　MT-R300
トレーニングテキスト　ステッピングモーター基板編

発行日　　2012-4-20　Rev1.01
発　行　　サンハヤト株式会社
〒 170-0005　東京都豊島区南大塚 3 丁目 40 番 1 号

©2011 Sunhayato Corp. All rights reserved.

SG11003